

Sviluppo



Di Michele Costabile



Cognito, ergo sum

Il mondo di oggi richiede app sempre più sofisticate, soprattutto se si tratta di applicazioni aziendali, o di giochi di successo. Non basta avere un'applicazione per una piattaforma mobile, occorre essere presenti almeno sui due sistemi operativi più diffusi e in fattori di forma diversi, dai tablet ai telefoni. Capita così che la stessa applicazione sia usata dallo stesso utente su un telefono Android, su un iPad e sul web.

Naturalmente gli utenti si aspettano di non perdere dati nel passaggio da un client all'altro. Trovarsi al quinto livello di un gioco sul telefono e al terzo sul tablet sarebbe considerato un segno di sciatteria. Questo richiede attenzione per gli sviluppatori che devono realizzare un servizio di archiviazione e sincronizzazione di dati e preferenze su un sito web. Inoltre, gestire tanti account diversi scocia e una buona parte degli utenti preferisce dirottare l'autenticazione verso siti che offrono la certificazione dell'identità, come Google, Yahoo!, Facebook, Twitter e Amazon.

Amazon ha creato un servizio per la gestione dell'identità e la sincronizzazione dello stato fra applicazioni, che si propone di offrire una soluzione pacchettizzata al problema di controllo dell'identità e dello stato. Il nome del servizio è Amazon Cognito, dall'ablativo dell'aggettivo

latino cognitus, cioè conosciuto. Cognito, che fa parte della collezione di servizi cloud di Amazon, probabilmente l'offerta più completa a disposizione degli sviluppatori, offre l'infrastruttura per creare identità, raggruppate per ruoli, gestire l'autenticazione o demandarla a server esterni OpenID e archiviare i dati volatili degli utenti in un semplice database basato su coppie formate da chiave e valore.

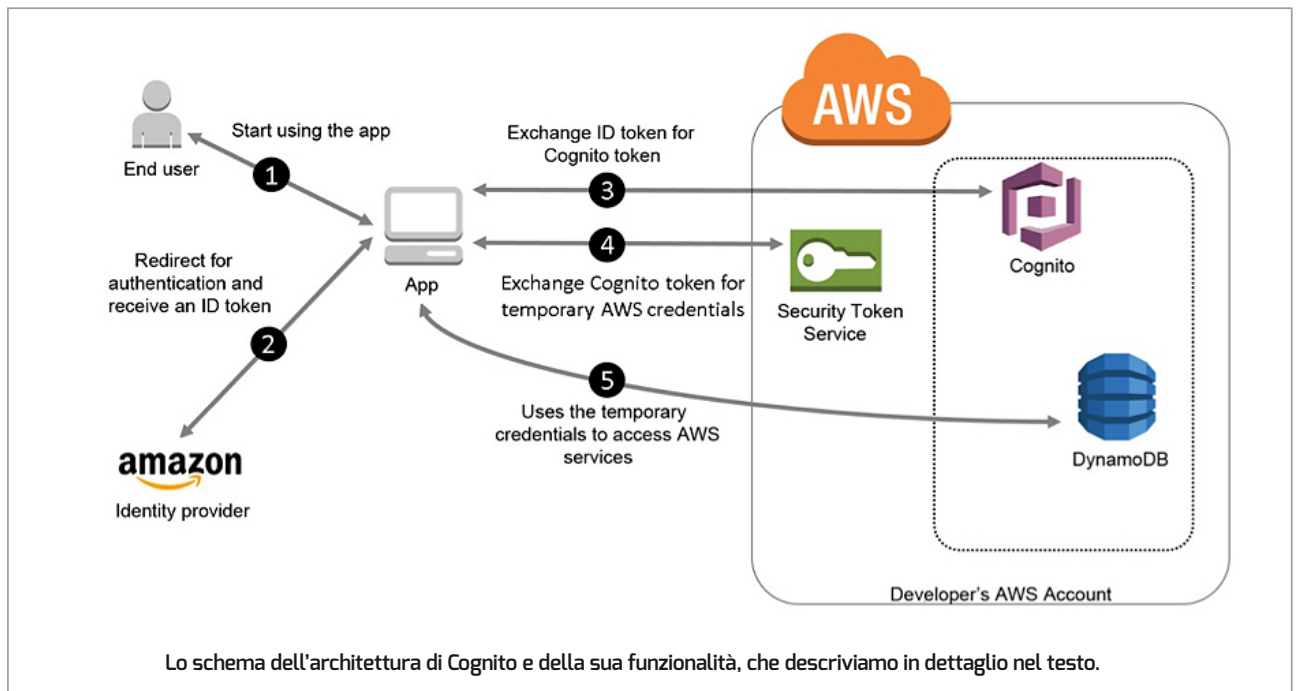
Il servizio di autenticazione degli utenti, che ne trasforma le credenziali in un identificativo unico, è gratuito a qualunque scala, mentre per lo spazio occupato dai dati e il numero di operazioni

Amazon presenta Cognito, un servizio che semplifica autenticazione e sincronizzazione dello stato fra device, consentendo di eseguire una app sul tablet e riprenderla sul telefono senza perdere dati.

di sincronizzazione, il tetto gratuito è rispettivamente di 10 Gbyte e un milione. Superato il tetto del servizio gratuito, i costi sono di quindici centesimi di dollaro ogni 10.000 operazioni di sincronizzazione e altrettanto per ogni GB di spazio occupato. Come nel caso degli altri servizi Amazon Web Services (AWS), quindi, è possibile sviluppare e avviare un'applicazione gratuitamente, iniziando a pagare bollette solo quando il traffico, e quindi il successo, lo giustifica.

COSA OCCORRE PER COMINCIARE

Per partire, è necessario prima di tutto aprire un account Amazon per sviluppatori, sul sito aws.amazon.com. Con le nostre credenziali di sviluppo possiamo creare una identity pool, ossia uno spazio per la configurazione



del servizio e il database degli utenti. Bisogna collegarsi all'indirizzo console. aws.amazon.com, selezionare il servizio Amazon Cognito, nell'area dei servizi mobili, e fare un clic sul pulsante Create new identity pool.

Alla nuova identity pool bisogna dare un nome, per esempio quello dell'applicazione, poi selezionare una checkbox per consentire, o meno, l'uso da parte di utenti non autenticati. Infine, nella sezione inferiore della pagina di creazione, possiamo selezionare un certo numero di servizi esterni, che vogliamo consentire di usare ai nostri utenti per identificarsi.

Possiamo scegliere fra Amazon, Facebook, Google+, Twitter e un servizio OpenID generico. Per ogni servizio c'è una maschera dedicata per i parametri di accesso al servizio di autenticazione, per esempio l'App ID per Amazon e Facebook, o la coppia formata da Consumer Key e Consumer Secret nel caso di Twitter.

Nel passo successivo, si possono creare i ruoli per gli utenti autenticati e i non autenticati nello Identity and Access Management di Amazon, che in sigla fa, opportunamente, IAM. Il servizio è gratuito e non è fatturato da Amazon, ma rientra nei costi degli altri servizi che usano l'infrastruttura.

Le possibilità di configurazione di IAM sono molto ricche e permettono di maneggiare a grana fine i diritti di accesso alle varie risorse del cloud per utenti e gruppi. Per approfondimenti rimandiamo alla documentazione specifica del

servizio (docs.aws.amazon.com/IAM/latest). La configurazione del servizio non richiede altro e la pagina successiva della console ci manda al link per scaricare lo sdk, disponibile sia per Android, sia per iOS, e esplorare la documentazione.

USARE IL SERVIZIO

Per usare il servizio bisogna scaricare lo sdk per Android o iOS dall'indirizzo aws.amazon.com/mobile/sdk, che comprende le librerie per tutti i servizi offerti da Amazon e la documentazione delle numerose funzioni, che è scaricata localmente in formato html e, nel caso del kit iOS, anche nel formato docset usato da Xcode. Dopo avere integrato nell'ambiente di sviluppo lo sdk, possiamo iniziare a usare il codice, partendo dall'interfaccia iniziale di Cognito, chiamata `CredentialsProvider`, che è l'inizio delle cose e viene messa in esercizio con questo codice Objective-C.

```
AWSCognitoCredentialsProvider
*credentialsProvider =
[AWSCognitoCredentialsProvider
credentialsWithRegionType:AWSRegionUSEast1
accountId:@"1234567890", //
id dell'account AWS
identityPoolId:@"us-east-1:xxxxxxxx-xxxx-xxxx-xxxx-xxxx",
unauthRoleArn:@"arn:aws:iam::xxxxxx:role/xxxx:role/NomeDelRuolo",
authRoleArn:@"arn:aws:iam::xxxxxx:role/xxxx:role/NomeDelRuolo"]
```

```
];
O questo codice Java, nel caso di un'applicazione Android
```

```
CognitoCachingCredentialsProvider
credentialsProvider = new
CognitoCachingCredentialsProvider(
getContext(), // recupera il
contesto
"1234567890", // id
dell'account AWS
"us-east-1:xxxxxxxx-xxxx-xxxx-xxxx-xxxx", // id della
identity pool
"arn:aws:iam::xxxxxx:role/
NomeDelRuolo", // ARN del ruolo
autenticato
"arn:aws:iam::xxxxxx:role/
NomeDelRuolo", // ARN del ruolo non
autenticato
Regions.US_EAST_1 // Regione
);
```

I parametri dell'inizializzazione sono la regione in cui è stata creata la identity pool, l'account dello sviluppatore su Aws. Seguono l'identificativo della identity pool e gli identificativi dei ruoli per l'utente autenticato e quello non autenticato, espressi nel formato in cui sono indicate le risorse gestite da Aws, chiamato Arn (Amazon Resource Name).

ARCHITETTURA

L'architettura del servizio è quella mostrata nell'immagine. Il primo passo è (figura 1) l'accesso dell'utente

Il primo passo per la configurazione del servizio è la creazione di una identity pool, lo spazio in cui saranno conservate le login degli utenti e il Datastore associato. Possiamo configurare servizi di autenticazione esterni, come Google+, Twitter e Facebook.

Il sistema richiede di definire almeno due ruoli, quello per gli utenti autenticati e gli utenti anonimi, se abbiamo consentito l'uso anonimo dell'applicazione.

Dopo avere creato la identity pool e impostato i ruoli siamo pronti per partire.

all'applicazione, che (figura 2) causa un redirect al servizio di autenticazione di Amazon, che usa le credenziali dell'utente e un eventuale servizio esterno per produrre un ID token. Questo, (figura 3) può essere scambiato con un token temporaneo Cognito, con il quale (figura 4) l'infrastruttura può procurarsi credenziali Aws, che (figura 5) possono essere usate per identificare l'utente e i suoi diritti di accesso a risorse nel cloud, come un database DynamoDB, uno dei tanti servizi di Aws.

Ecco il codice con cui si inizia la transazione, nel caso di iOS

```
// Recupera il token da un identity provider.
```

```
NSString *token =
    "Token_From_Identity_Provider";
// Facebook
credentialsProvider.logins = @{
    AWSCognitoLoginProviderKeyFacebook:
    token };
// Google
credentialsProvider.logins = @{
    AWSCognitoLoginProviderKeyGoogle:
    token };
// Amazon
credentialsProvider.logins = @{
    gniToLoginProviderKeyLoginWithAmazon:
    token };
```

e in versione Android

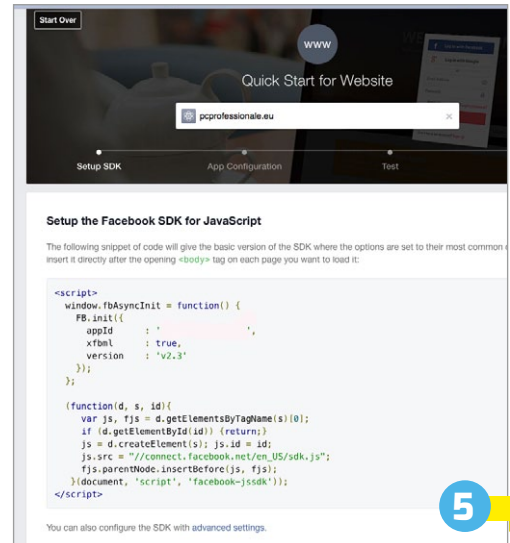
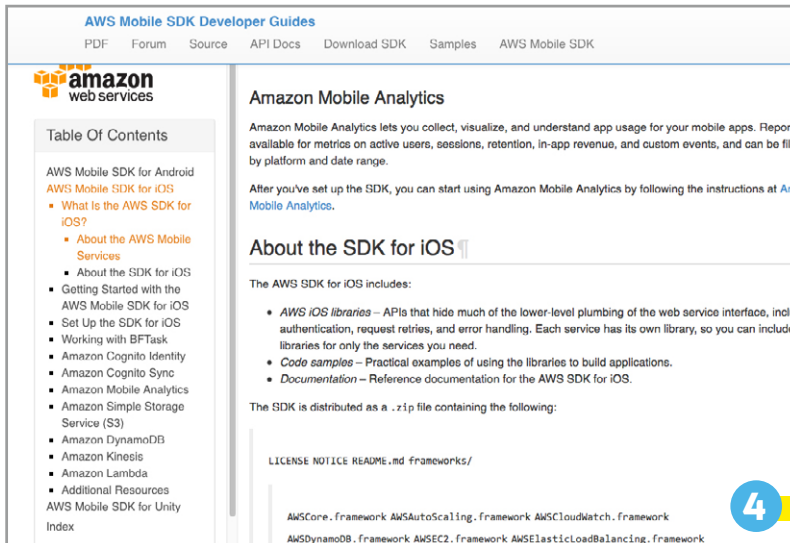
```
String token =
```

```
"Token_From_Identity_Provider";
Map logins = new HashMap();
// Facebook login token
logins.put("graph.facebook.com",
    token);
// Google login token
logins.put("accounts.google.com",
    token);
// Amazon login token
logins.put("www.amazon.com",
    token);
```

```
credentialsProvider.
withLogins(logins);
```

Il passaggio successivo creare un oggetto che rappresenta la configurazione dello sdk e iniziarlo con il credentialsProvider che abbiamo ottenuto nel passaggio precedente. Una volta costruito un oggetto AWSServiceConfiguration, potremo usare tutti i servizi cloud di Aws, quindi anche usare lo spazio di memorizzazione gestito da Cognito e i servizi di sincronizzazione dello stato dell'applicazione fra device diversi.

```
AWSServiceConfiguration
*configuration =
    [AWSServiceConfiguration
        configurationWithRegion:AWSRegionUSEast1
        provider:credentialsProvider];
// impostiamo questa configurazione
come default per tutti i servizi
[AWSServiceManager
    defaultServiceManager].
defaultServiceConfiguration =
    configuration;
```



Lo Sdk di amazon consente l'uso di tutti i servizi cloud e ha una ampia documentazione, disponibile in diverse lingue, ma non in Italiano.

Per usare Facebook per l'autenticazione, occorre richiedere un App ID, che consente di usare il servizio.

Questa operazione è molto più semplice nel caso di Android, dato che possiamo passare direttamente un `CredentialsProvider` al costruttore del client di un servizio, per esempio `DynamoDB`, come segue.

```
AmazonDynamoDB client = new AmazonDynamoDBClient(credentialsProvider);
```

SINCRONIZZARE I DATI

Vediamo, infine, come sincronizzare lo stato di un'applicazione fra device diversi, usando il `Datastore` e i servizi di sincronizzazione di `Cognito`. Per iniziare dobbiamo creare l'interfaccia verso il servizio, con il codice Objective-C seguente

```
AWSCognito *syncClient = [AWSCognito defaultCognito];
```

O con questo codice Java su Android

```
CognitoSyncManager syncClient = new CognitoSyncManager(myActivity.getContext(), COGNITO_POOL_ID, Regions.US_EAST_1, cognitoProvider);
```

Ottenuto un `syncClient`, possiamo inviargli il messaggio di creare o aprire un `dataset`, quindi impostare dei dati, per esempio, corrispondenti al nome, livello e punteggio di un giocatore come segue:

```
AWSCognitoDataset *dataset = [syncClient openOrCreateDataset:@"GameInfo"];
```

```
// ci sincronizziamo una volta, dopo l'apertura
[dataset synchronize];
```

```
// impostiamo alcuni dati
[dataset setString:@"Michele" forKey:@"playerName"];
[dataset setString:@"3" forKey:@"currentLevel"];
[dataset setString:@"120345" forKey:@"highScore"];
```

```
// inviamo i dati per la sincronizzazione
[dataset synchronize];
```

Ecco lo stesso codice in versione Android

```
Dataset dataset = client.openOrCreateDataset("GameInfo");
```

```
// recuperiamo gli ultimi dati dal cloud
dataset.synchronize(this, syncCallback);
```

```
// impostiamo alcuni dati
dataset.put("playerName", "Michele");
dataset.put("currentLevel", "3");
dataset.put("highScore", "120345");
```

```
// sincronizziamo
dataset.synchronize(this, syncCallback);
```

Adesso, su qualsiasi device possiamo recuperare questi dati dal `Datastore`, come segue

```
NSString *playerName = [dataset valueForKey:@"playerName"];
```

```
NSString *currentLevel = [dataset valueForKey:@"currentLevel"];
NSString *highScore = [dataset valueForKey:@"highScore"];
```

In versione Android

```
String playerName = dataset.get("playerName");
String currentLevel = dataset.get("currentLevel");
String highScore = dataset.get("highScore");
```

CONCLUSIONI

I servizi cloud di Amazon richiedono un po' di investimento di tempo. Come con ogni tecnologia, bisogna prevedere uno scalino iniziale per l'apprendimento e la configurazione. Sull'altro piatto della bilancia, però, abbiamo una contropartita interessante, in questo caso un aspetto professionale.

Volendo realizzare un'applicazione in versione web e mobile, o su più piattaforme mobili, gli utenti potrebbero avere poca comprensione per uno schema di autenticazione che non offra la possibilità di riutilizzare credenziali e per la mancanza di un sistema di sincronizzazione dello stato e delle preferenze. Considerando che dobbiamo anticipare solo un investimento di tempo per usare i servizi di `Aws`, confidando nell'ampio tetto per l'uso gratuito, l'unico ostacolo che ci possiamo aspettare è solo il tempo, ma ci pare di avere mostrato che il servizio ha una logica e un'architettura che lo rendono piuttosto semplice da usare.